REMARKS

Applicant thanks the Examiner for total consideration given the present application. Claims 1, 4, 7-10, 12, 14, 16, 18, 20, and 21 are currently pending, of which claims 1, 9, 16, and 18 are independent. Claims 1, 9, 16, and 18 have been amended through this Reply. Applicant respectfully requests reconsideration of the rejected claims in light of the remarks presented herein, and earnestly seeks timely allowance of all pending claims.

Claim Objection

Claims 1, 9, 16 and 18 are objected to because of informalities. These claims have been amended to address this issue. Accordingly, it is respectfully requested to withdraw this objection.

Rejection Under 35 U.S.C. § 112

Claims 1, 9, 16 and 18 stand rejected under 35 U.S.C. § 112, first paragraph, for allegedly containing subject matter not described in the Specification in such a way as to reasonably convey to one skilled in the art that the inventor had possession of the claimed invention at the time of filing. Applicants respectfully traverse this rejection.

Applicants point out that MPEP § 2163 sets forth guidelines for the examination of patent applications under the "Written Description" requirement of 35 U.S.C. § 112, first paragraph. Specifically, the second paragraph of MPEP § 2163.I.B indicates that the requirement for the Specification to support added claim limitations is not an in *haec verba* requirement (i.e., the Specification is not required to use the exact language in the claims). Instead, this section of the MPEP indicates that the Specification may support added claim limitations through express, implicit, or inherent disclosure.

Furthermore, MPEP § 2163.II.A lists the methodology for the Examiner to follow in order to determine the adequacy of the Written Description. This methodology includes the following steps:

1. For each claim, determine what the claim as a whole covers;

2. Review the entire application to understand how Applicants provide support for the claimed invention including each element and/or step; and

3. Determine whether there is sufficient Written Description to inform a skilled artisan that Applicants were in possession of the claimed invention as a whole at the time the application was filed.

Applicants respectfully submit that the Examiner did not follow this methodology in rejecting the claims. Instead, it appears that the Examiner concluded that a particular claim element, i.e., "type safe and non-type safe values and access to such values," is not enabled because the exact language is not found in the Specification. Applicants respectfully submit that such analysis is not permitted according to the aforementioned methodology required by the MPEP.

Furthermore, Applicants submit that the claimed "type safe" and "non-type safe" features are impliedly described in the Specification and Figures. For example, in Fig. 2 of the instant Specification, illustrates that that model element 204 is a base class for all of the data that is located in store 202 wherein model element 204 implements constructs as described by the meta-data. Further, the meta-data contains information used to ***describe other data such as type***, name, default values, or restrictions imposed on the data. (See also paragraph [0027].) Further, paragraph [0034] the instant Specification describes that a first subclass of a singleton model field handler object 402 is a ***typed*** model element field handler object 406. Based on the foregoing, type-safe refers to the operations that can be performed on data in a language sanctioned by the type of the data. Thus, it is clear to one of ordinary skill in the art that such "type safe" and "non-type safe" features are impliedly described in the specification and figures.

Accordingly, Applicant respectfully submits that those of ordinary skill in the art would immediately recognize that the claimed "type safe" and "non-type safe" features are described in the Specification so as to convey that the inventor had possession of the claimed invention at the time of filing. Although Applicants do not necessarily agree with the Examiner regarding this rejection, claims 1, 9, 16, and 18 have been amended merely to expedite prosecution. Applicant again wishes to remind the Examiner that there is no requirement that the Specification use the exact language of the claims.

Therefore, Applicant respectfully requests reconsideration and withdrawal of this rejection.

Rejection Under 35 U.S.C. § 103

The Examiner rejects claims 1, 4, 7-10, 12, 14, 16, 18 and 20-21 under 35 U.S.C. § 103(a) as allegedly being unpatentable over Wall et al. (U.S. Publication No. 2002/0087557)[hereinafter "Wall"] in view of Matula et al. (U.S. Publication No. 2002/0165995)[hereinafter "Matula"] and further in view of Coad et al. U.S. Patent No. 6,851,105)[hereinafter "Coad"]. Applicant respectfully traverses this rejection.

For a Section 103 rejection to be proper, a *prima facie* case of obviousness must be established. *See M.P.E.P. 2142.* One requirement to establish *prima facie* case of obviousness is that the prior art references, when combined, must teach or suggest all claim limitations. *See M.P.E.P. 2142; M.P.E.P. 706.02(j).* Thus, if the cited references fail to teach or suggest one or more elements, then the rejection is improper and must be withdrawn.

In this instance, in regard to claims 1 and 16, it is respectfully submitted that none of the cited prior art references, alone or in combination, teaches or suggests, *inter alia*,

> "a model element class configured to implement the constructs described by metadata; ***the model element class storing an attribute value directly in a private member field of the model element class in a memory block where a declaring class is stored*** . . .
>
> ***a base field handler class*** which acts as an ***intermediary generic mechanism*** when getting the field values in the model element field handler object, wherein the get value function is configured to:
>
> i) ***dispatch the field values to a first sub-classed get field value method*** in the first subclass ;

ii) access the model element class and return the attribute value stored in the private member field of the model element class ***directly*** upon request; and

iii) ***provide an entry point for other sub-classed get field value method representing a plurality of types so that general purpose client code can access the field values irrespective of the type of the first sub-classed get field value***" (*emphasis added.*)

In regard to claims 9 and 18, it is respectfully submitted that none of the cited prior art references, alone or in combination, teaches or suggests, *inter alia*,

"a model element class configured to implement the constructs described by metadata; ***the model element class storing an attribute value directly in a private member field of the model element class in a memory block where a declaring class is stored*** . . .

a base field handler class which acts as ***an intermediary generic mechanism*** when setting the field values in the model element field handler object, wherein the set value function is configured to:

i) provide validation of a new value; and

ii) record necessary undo information associated with the new value;

iii) ***dispatch the field values to a first sub-classed set field value method*** in the first subclass;

iv) access the model element class and return the attribute value stored in the private member field of the model element class ***directly*** upon request; and

v) ***provide an entry point for other sub-classed set field value method representing a plurality of types so that general purpose client code can access the field values irrespective of the type of the first sub-classed set value***" (*emphasis added.*)

The claimed invention is directed to efficiently implementing object model attributes. There are several drawbacks in conventional implementation of object model attributes. For example, one drawback of conventional meta-models is that large amounts of memory overhead and computer processing time are required to store, find, and manipulate boxed attribute values within these models. Another drawback of the conventional meta-models is directed to debugging issues. For example, since in the conventional meta-models, values such as the boxed attribute values typically end up deeply nested within wrapper classes, debugging of generated codes becomes extremely difficult. Further drawback in the conventional meta-models is that complex calling conventions are required to support complicated features or tool, such as undo/redo and notifications which are implemented in many applications.

In order to address the above-mentioned drawbacks, the claimed invention requires, among other features, a base field handler class that acts as ***an intermediary generic mechanism*** when ***setting*** and ***getting*** the field values in the model element object.

The claimed get value function is configured to ***dispatch the field values to a first sub-classed get field value method*** in the first subclass and access the model element class and return the attribute value stored in the private member field of the model element class ***directly*** upon request. The get value function is also configured to ***provide an entry point for other sub-classed get field value method representing a plurality of types so that general purpose client code can access the field values irrespective of the type of the first sub-classed get field value***.

The claimed set value function provides analogous access as get value, but additionally provides validation of a new value, records necessary undo information associated with the new value and dispatches to a sub-classed set field value method in the first subclass.

After careful review of the applied prior art references, Applicant respectfully submits that neither the cited portions nor any other portions of Wall, Matula, and Coad teach or suggest the above-identified claim features of claims 1, 9, 16, and 18.

As previously submitted, Wall merely discloses a conventional method and apparatus for providing access control for a decentralized or emergent model on a computer network in which a model is defined using hierarchical relationship of servers, models, and objects.

Matula, on the other hand, discloses a method for dynamic implementation of a Java Metadata Interface (JMI) to a metamodel in which JMI interfaces are implemented as subclasses of handler classes.

Coad is directed to generating, applying, and defining patterns for software development. Although Coad discloses a Singleton pattern in generating a pattern instance, Coad provides no specifics about any base field handler class which acts as an intermediary generic mechanism when setting and getting the field values in the model element field handler object nor does Coad teach or suggest the above-identified claimed configuration of the "get value function" and the "set value function".

In addition, it is respectfully submitted that Wall fails to teach or suggest a model element class configured to implement the constructs described ***by metadata a model element class storing an attribute value directly in a private member field of the model element class in a memory block where a declaring class is stored***. The Examiner relies on paragraph [0064] of Wall as disclosing this feature. More specifically, the Examiner alleges that the "constructs described by metadata" is a class declaration of how each attribute function, i.e., integer, string, etc. (*See page 5, section a. of the Office Action.*)

First, it is respectfully submitted that the entire reference is silent on any "***metadata***", let alone "a model element class that is configured to implement constructs described by metadata."

Second, nowhere does Wall teach or suggest ***a model element class storing an attribute value directly in a private member field of the model element class in a memory block where a declaring class is stored***. The Examiner's mere allegation that a memory block could be a hard drive, a database, etc., has no logical basis for the claimed element. The Examiner has failed to provide any teaching or suggestion from the cited reference that attribute value is stored directly in a private member field of a model element class in a memory block where a declaring class is stored. The relied upon section of Wall merely discloses that object classes 500 and 501 are example classes that define multiple attributes/fields (502-512) used for describing the state and behavior of a model. Again, nowhere does Wall teach or suggest a model element class storing attribute value directly in a private member field of a model element class in a memory block where a declaring class is stored.

Therefore, for at least these reasons, it is respectfully submitted that independent claims 1, 9, 16, and 18 are allowable over Wall, Matula, and Coad. Claims 4, 7, 8, 10, 12, 14, 20, and 21 are at least allowable by virtue of their dependency on corresponding allowable independent claim.

CONCLUSION

In view of the above amendment, Applicant believes the pending application is in condition for allowance.

Should there be any outstanding matters that need to be resolved in the present application, the Examiner is respectfully requested to contact Ali M. Imam Reg. No. 58,755 at the telephone number of the undersigned below, to conduct an interview in an effort to expedite prosecution in connection with the present application.

If necessary, the Commissioner is hereby authorized in this, concurrent, and future replies to charge payment or credit any overpayment to Deposit Account No. 02-2448 for any additional fees required under 37.C.F.R. §§1.16 or 1.147; particularly, extension of time fees.

Dated: April 21, 2009

Respectfully submitted,

By [illegible] #58,755
for Michael K. Mutter
Registration No.: 29,680
BIRCH, STEWART, KOLASCH & BIRCH, LLP
8110 Gatehouse Road
Suite 100 East
P.O. Box 747
Falls Church, Virginia 22040-0747
(703) 205-8000
Attorney for Applicant